工事立会者手配問題に対する集合被覆アプローチ

NTTデータ数理システム 学術奨励賞 2021

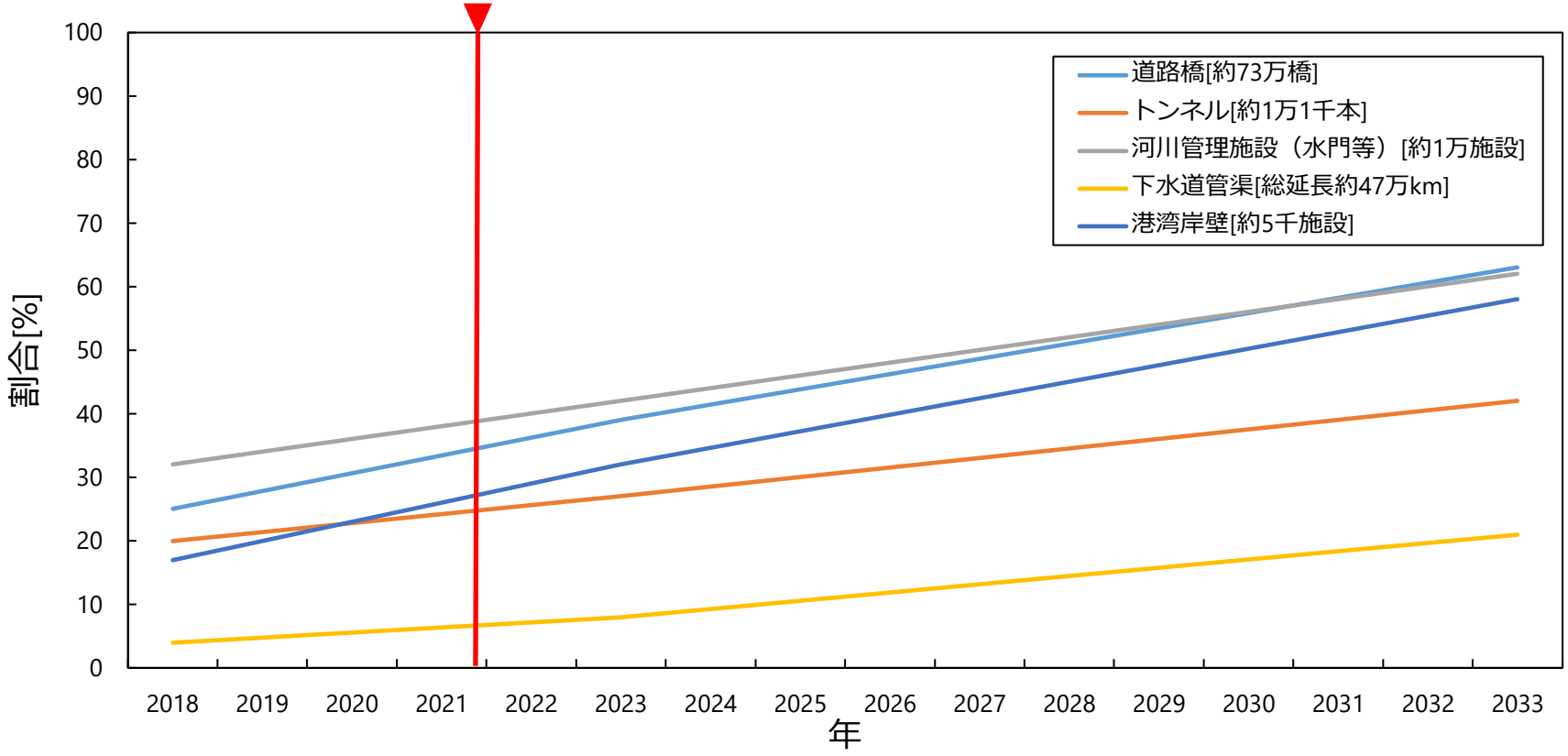
2021年12月3日

名古屋大学/西日本電信電話株式会社　高須賀将秀
静岡大学　呉偉
名古屋大学　柳浦睦憲

目次

1．はじめに：研究の背景 インフラ老朽化について

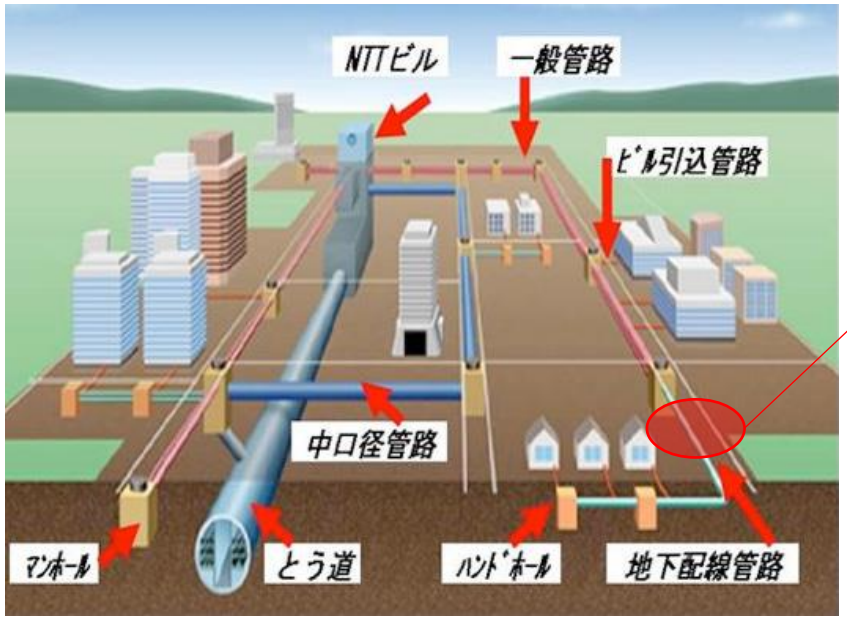
- 近年道路等のインフラの老朽化が深刻な社会問題となっている
- インフラを補修する工事等の件数が増加している



建設後５０年以上経過する社会資本の割合 (国土交通省)

1．はじめに：研究の背景 インフラの補修工事について

- インフラの補修工事には各設備に関する知識を有する監督者が必要である[1, 2]
- 電話等の通信設備への影響を防ぐためにNTTの社員が立会を行っている[3, 4, 5]

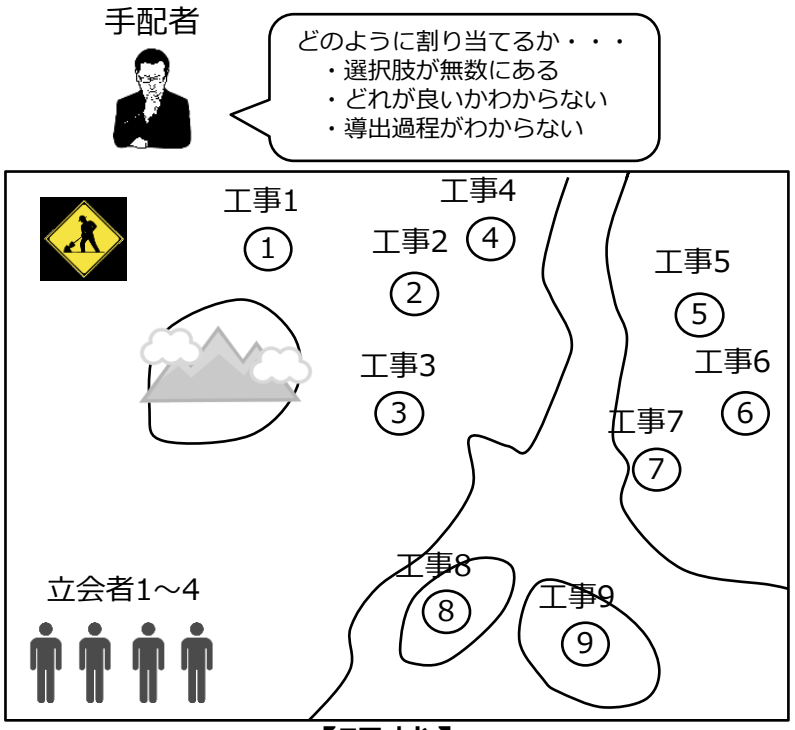


NTT管路系設備

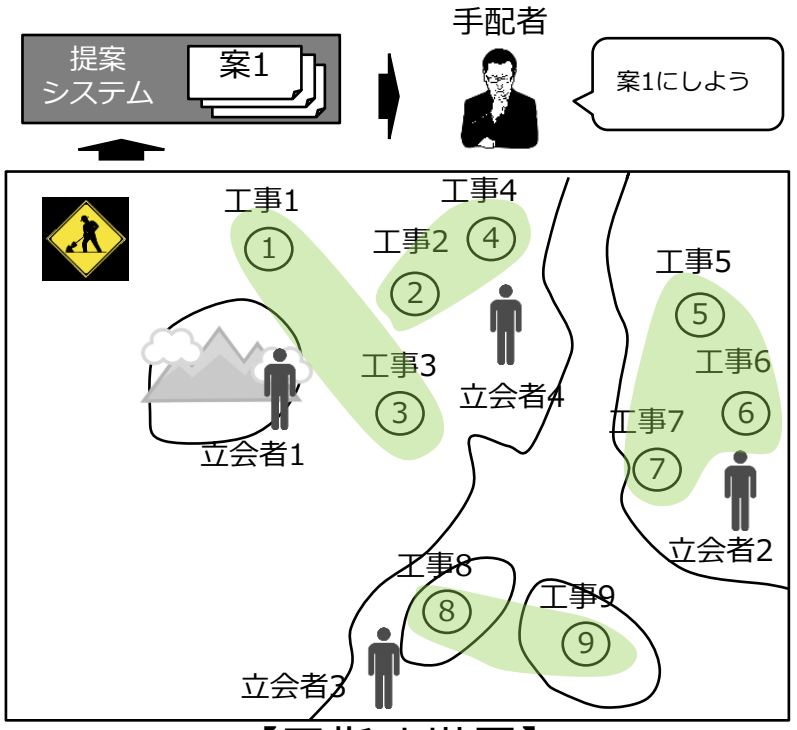
道路の工事（掘削）の様子

1．はじめに：研究の背景 工事割当の現状と目指す世界

- 複数の工事に立会者を割り当てる人を“手配者”と呼ぶ
- 手配者は割当に関する高度なスキルやノウハウを有している
- 割当作業 (手配)は判断基準も不明確であることが多い



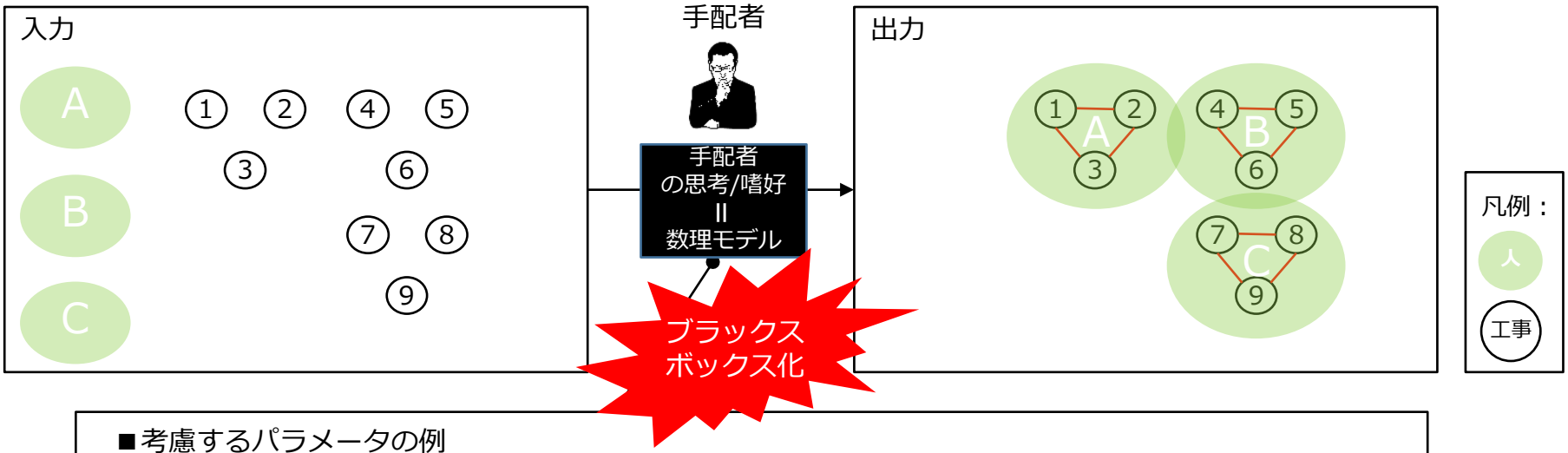
【現状】



【目指す世界】

2－1．工事立会者手配問題について（1/2）

- 与えられた全ての工事に対して立会者の割当を決定する問題である
- 移動距離や立会者のスキル等の様々な条件を考慮し割当を決定している
- 考慮すべき条件は万人共通の条件もあれば手配者の思考や嗜好によって異なるものもある



■考慮するパラメータの例

パラメータ	説明
稼働量	1日に担当する工事数
移動時間	工事間の移動時間
工事の難易度	工事の種別や規模から決まる値
立会者のスキル	立会者が所持しているスキル 通常工事ごとに異なる値
工事の開始時刻	工事に要請される開始時刻

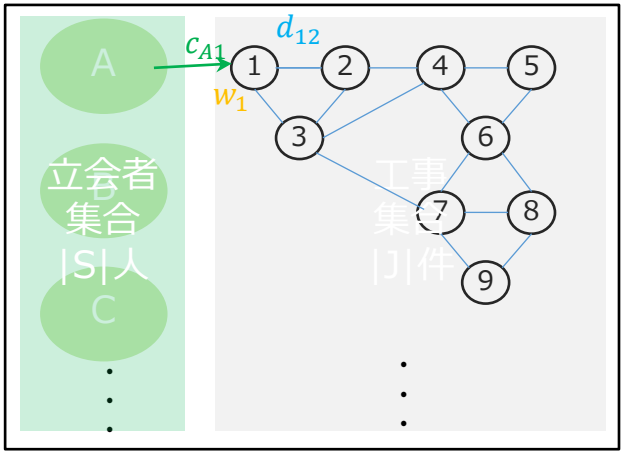
2－1．工事立会者手配問題について（2/2）

■ 入力は以下の通り
- 立会者の集合S および工事の集合J
- 工事k からl への移動時間d_{kl}
- 立会者s に工事k を割り当てたときの割当ペナルティc_{sk}
- 工事の難易度w_k
- 各立会者に割り当てられた工事の難易度の和に対する上限W

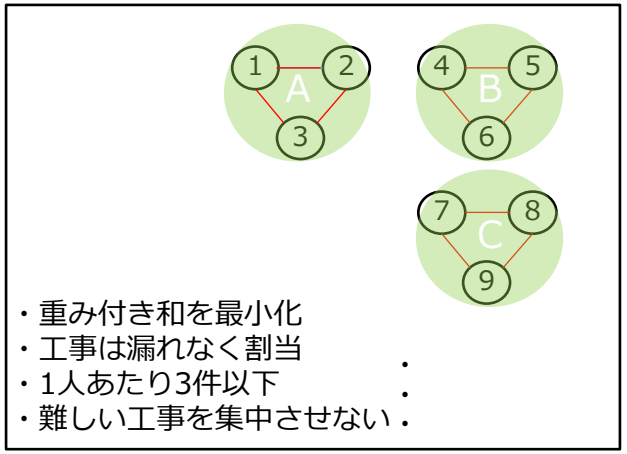
■ 制約条件は以下の通り
- 各工事にちょうど1 人の立会者を割り当てること
- 各立会者に割り当てられる工事数は3 件以下とすること
- 各立会者に割り当てられる工事の総難易度はW 以下とすること

■ すべての立会者の総移動時間と総割当てペナルティ（工事に対する立会者のスキルを示す指標）の重み付き和を最小化することが目的である

■入力



■出力



2－2．集合被覆アプローチによる数理モデル

- 現場の手配者にヒアリングを繰り返し行い，構築した数理モデルは以下である
- 以下の数理モデルは一般化上界制約付き集合被覆問題と呼ばれる問題であり[6]，本研究では工事立会者手配問題をその問題に定式化した
- 以下を満たす実行可能なルートを列挙し全ての工事を被覆するルートを選ぶ
  - A) $|J|$ 件の工事に対し，$|S|$ 人の立会者を割り当てる
  - B) 各工事にちょうど1人の立会者を割り当てる
  - C) 各立会者に割り当てられる工事数は3 件以下とする
  - D) 各立会者に割り当てられる工事の総難易度はW 点以下とする
  - E) すべての立会者の総移動時間と総割当ペナルティの重み付き和を最小化する

❑ 集合被覆アプローチによる数理モデル

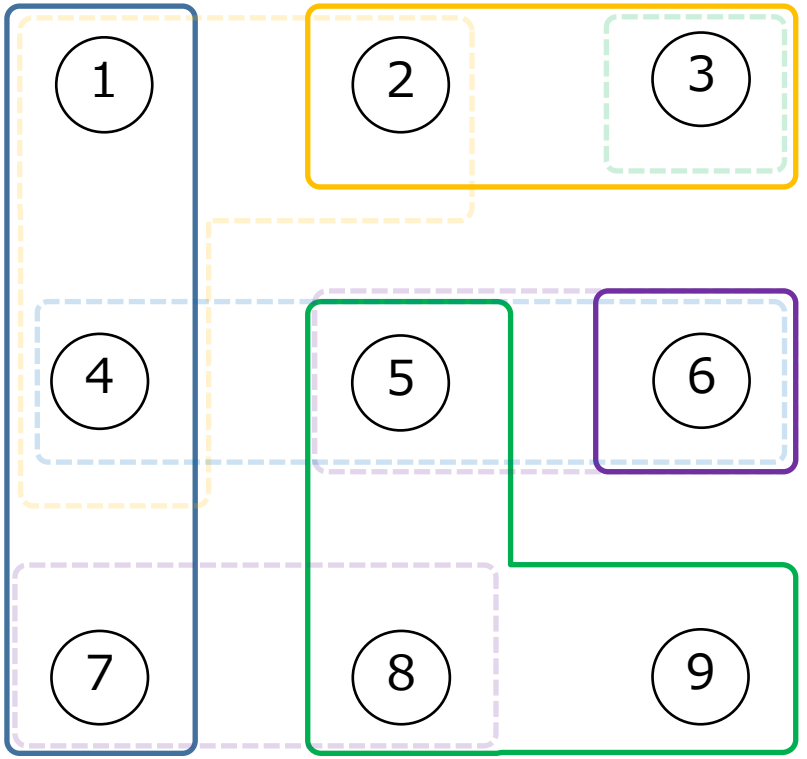
$$\min \sum_{r\in R}\sum_{s\in S}\left(\tilde{d}_r + \alpha\tilde{c}_{rs}\right)y_{rs}$$

$$\text{s.t.} \quad \sum_{r\in R}\sum_{s\in S} a_{kr}\, y_{rs} \geq 1 \quad (\forall k \in J)$$

$$\sum_{r\in R} y_{rs} \leq 1 \quad (\forall s \in S)$$

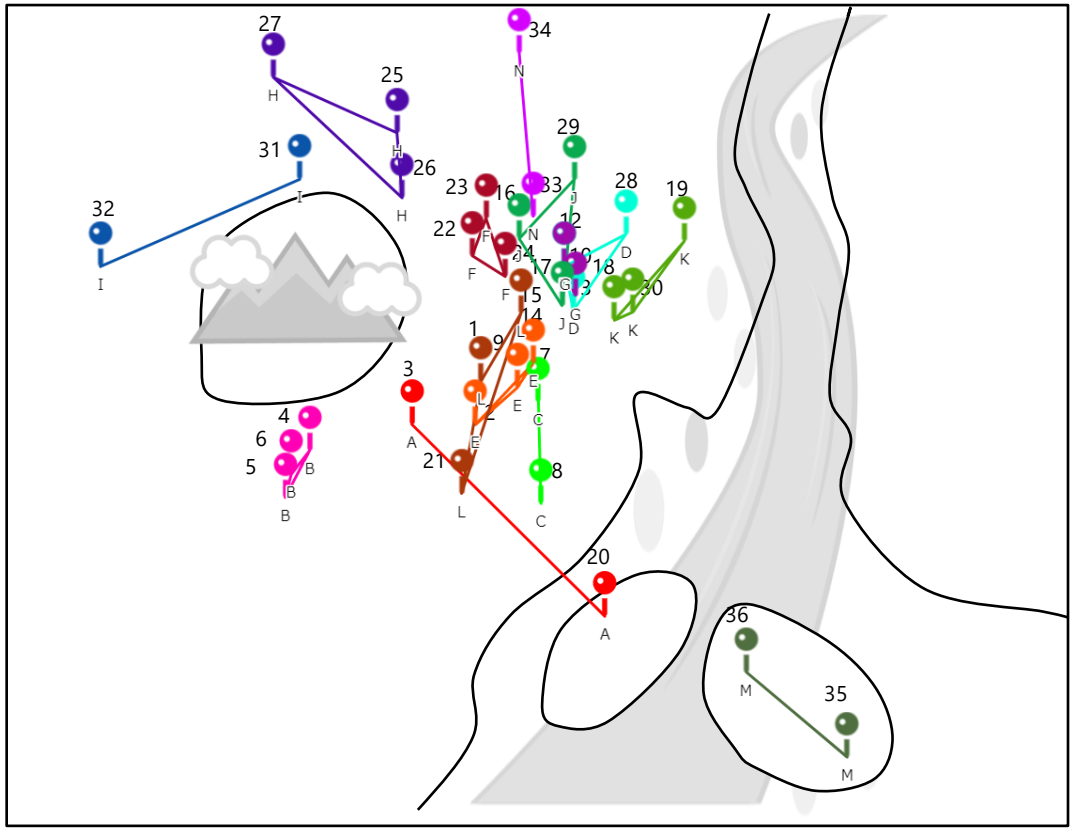
$$y_{rs} \in \{0,1\} \quad (\forall r \in R, \forall s \in S)$$

S： 立合者の集合
J： 工事の集合
R： 実行可能なルートの集合
$\tilde{d}_r$: ルートrの合計移動時間
$\tilde{c}_{rs}$: 立会者sがルートrを担当したときの合計割当ペナルティ
a_{kr}: ルートrが工事kを含むとき$a_{kr}=1$，含まないとき$a_{kr}=0$
α：総移動時間と総割当ペナルティの重み係数



2－3．実データを入力したときの計算結果

- 数理モデル ($\alpha = 50$)により出された解を以下に示す
- 本結果は現場の手配結果として概ね実用的であるという意見を得ている
- この結果は手配者の手配結果と工事のグループ分けが類似している
- 総移動時間は手配者の手配結果より1 % 程度多いもののほぼ同等であり総割当ペナルティは手配者の手配結果の約半分となっている

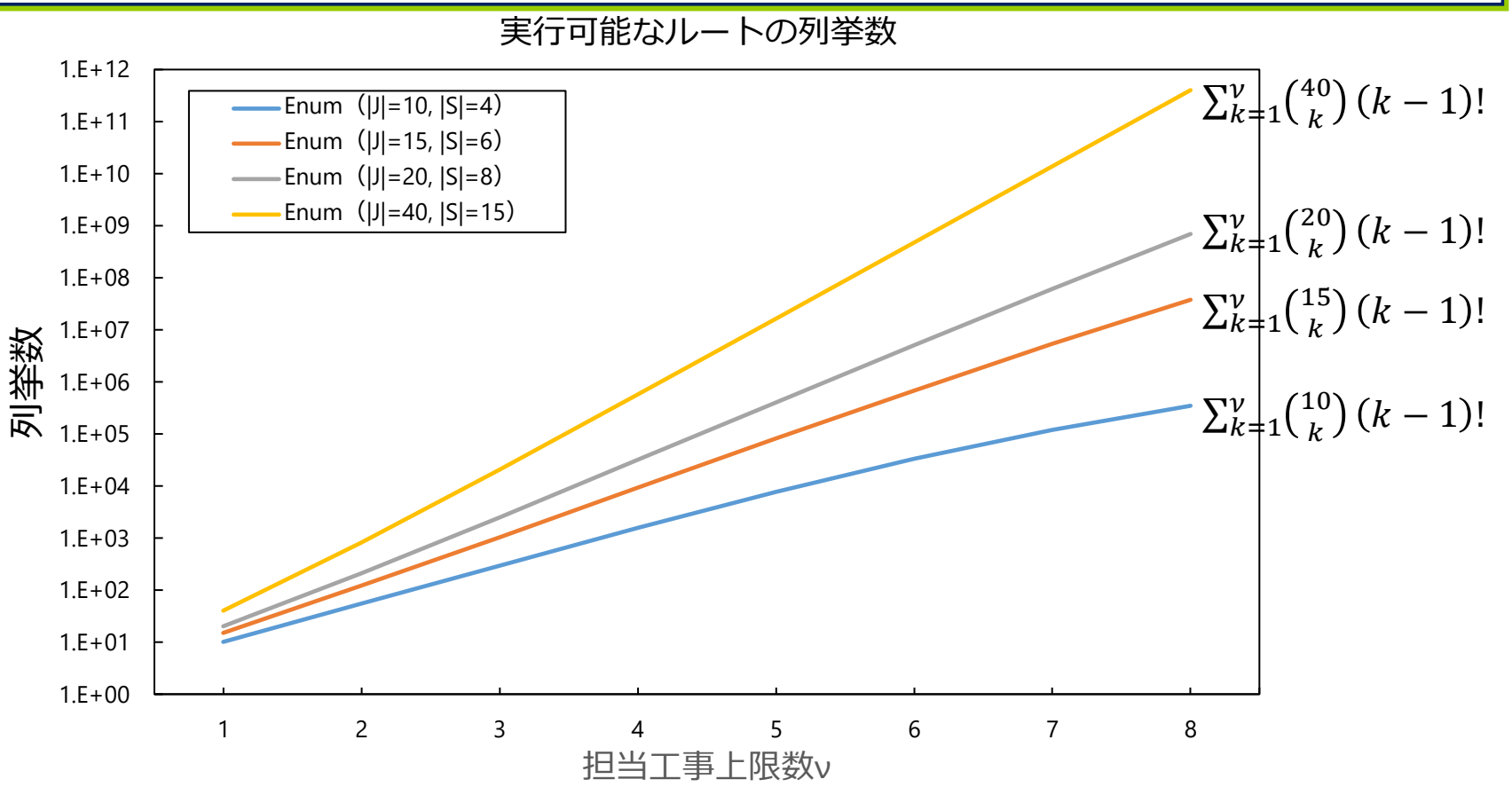


数理モデル （$\alpha = 50$）における結果

立会者	T1	T2	T3	合計移動時間	合計割当ペナルティ	合計難易度
A	3	20	-	1,310	4	6
B	6	4	5	518	7	8
C	8	7	-	676	3	5
D	10	12	28	816	9	7
E	9	14	2	595	11	8
F	23	24	22	604	6	8
G	13	11	-	446	5	5
H	27	25	26	1,115	10	7
I	31	32	-	1,276	2	5
J	29	17	16	928	6	8
K	30	18	19	725	8	8
L	15	1	21	1,086	5	6
M	36	35	-	702	2	5
N	34	33	-	796	3	5
			総	11,593	81	91

2－4．新しい要望

- 数理モデルによる手配結果を実業務で運用したところ，立会者の担当する工事件数は3 件以下ではなく4 件以上割り当てたいという新しい要望があがった
- 4 件以上となる実行可能なルートを全て列挙すると列挙数が指数的に増加するため，先行研究[7, 8] での集合被覆アプローチ による実行可能なルートを全列挙する方法（Enum）は実用的でない



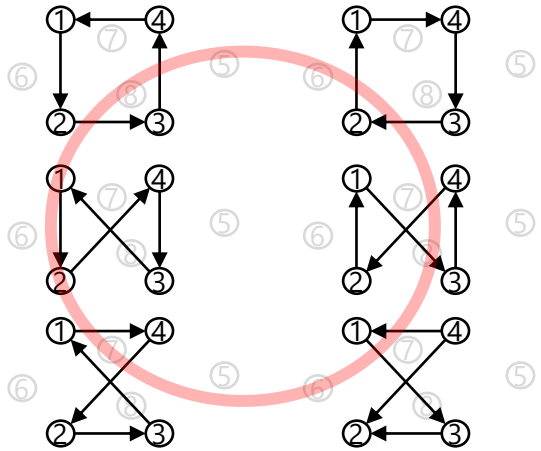
3．集合被覆モデルの改善（1/2）

- Enumでは立会者の担当する工事件数を増加すると実行可能なルートの列挙数が指数的に増加する
- 集合被覆アプローチを用いた定式化に基づき，ルートを列挙するときに巡回セールスマン問題として定式化[9] し，各立会者が担当する工事の部分集合の候補の各々に対して最適な巡回路のみを選ぶことで，列挙数を削減する（Enum+）

❑ Enum

実行可能なルートを全て列挙する

列挙数 $\sum_{k=1}^{\nu}\binom{|J|}{k}(k-1)!$ 個

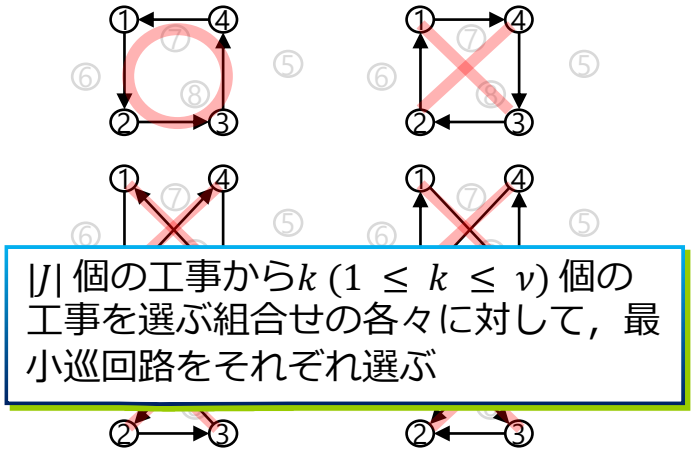


集合被覆アプローチを用いた定式化で解く

❑ Enum+

最適な巡回路のみを列挙する

列挙数 $\sum_{k=1}^{\nu}\binom{|J|}{k}$ 個

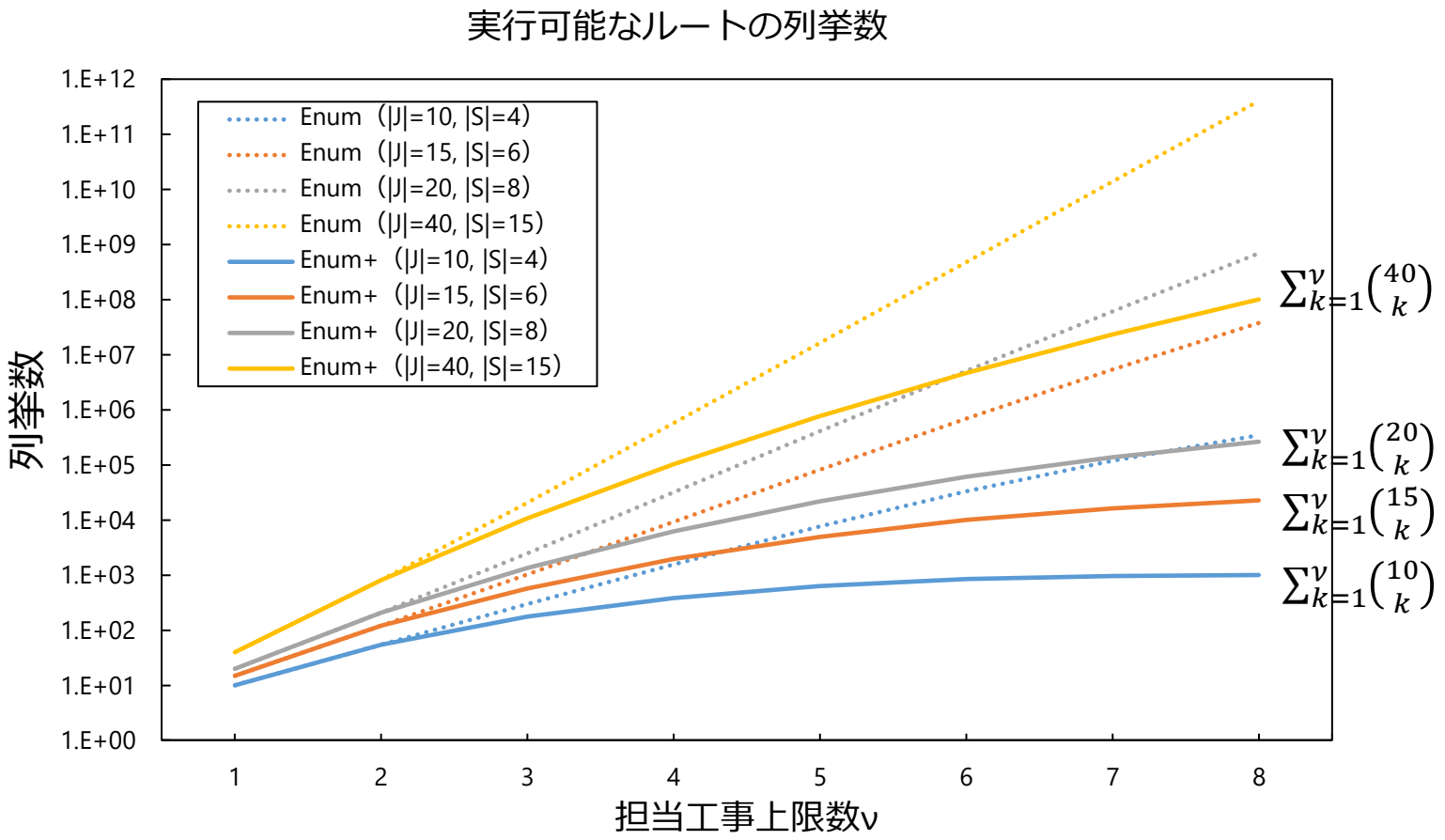


$|J|$ 個の工事からk $(1 \leq k \leq \nu)$ 個の工事を選ぶ組合せの各々に対して，最小巡回路をそれぞれ選ぶ

集合被覆アプローチを用いた定式化で解く

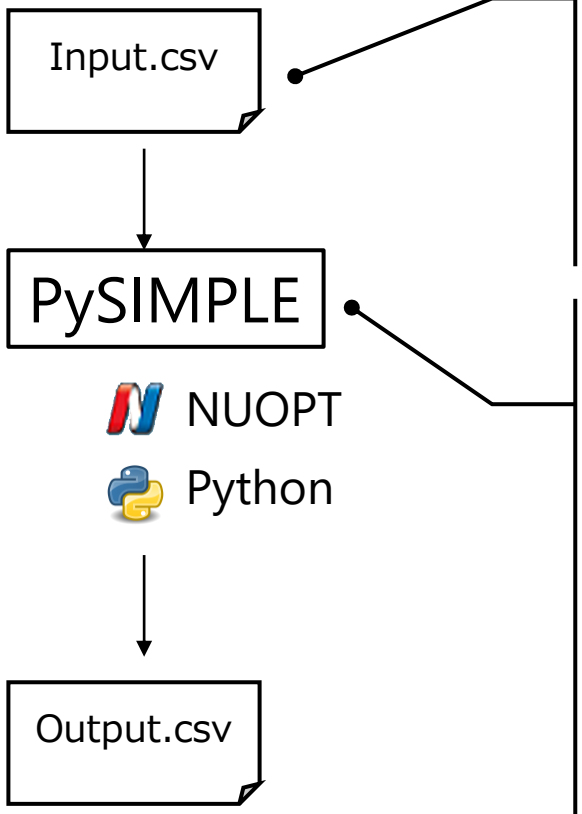
3．集合被覆モデルの改善（2/2）

- EnumとEnum+に対する実行可能なルートの列挙数を比較した結果を以下に示す
- EnumよりEnum+のほうが実行可能なルートの列挙数が削減している



4．実装（1/2）

- Enum+はEnumに比べて実装が複雑になるため，PySIMPLEを用いる
- PySIMPLEでの簡易な実装コードを以下に示す



S ：立会者の集合
J：工事の集合
d_{kl}：工事kからlへの移動時間
c_{ks}：立会者sに工事kを割り当てたときの割当ペナルティ
w_k：工事kの難易度
W：立会者に割り当てられた工事の難易度の和
α：目的関数内の重み係数
v：各立会者に割り当てられる工事数の上限

```
from pysimple import *

def read_file(file_name):                    #Inputファイルの読み込み
def solve_tsp(tour, d_kl):                   #集合tourの最短巡回路
def enum_cols(S, J, d_kl, c_sk, w_k, W, v):  #列挙
    for n in range(2, v + 1):
        for tour in itertools.combinations(J, n):
            solve_tsp(tour, d_kl)
def solve_enum(cols, S, J):                  #集合被覆モデル

if __name__ == '__main__':
    read_file("./input.csv")
    cols = enum_cols(S, J, d_kl, c_sk, w_k, W, v)
    solve_enum(cols, S, J, α)
```

4．実装（2/2）

■ def solve_tsp関数とdef solve_enum関数の簡易な実装コードを以下に示す

```
def solve_tsp(tour, d_kl):

    k = Element(value=tour)
    l = Element(value=tour)
    k' = k != l.set[0]
    k_ = Element(set=k'.set)
    l_ = Element(set=k'.set)
    kl = k != l
    kl_ = k_ != l_
    num = len(k'.set)
    d = Parameter(index=(k, l), value= d_kl)

    # 変数
    x = BinaryVariable(index=(k, l))
    y = Variable(index=k', lb=1, ub=num)

    # 目的関数
    problem += Sum(x[kl]*d [kl])

    #制約条件
    problem += Sum(x[kl], kl(1)) == 1
    problem += Sum(x[kl], kl(0)) == 1
    problem += y[kl_(0)] - y[kl_(1)] + num*x[kl_]
                <= num-1

    problem.solve
```

```
def solve_enum(cols, S, J, α):

    s = Element(value=S.keys())
    k = Element(value=J.keys())
    c = Element(value=cols.keys())
    d = Parameter(index=c, value= d̃_c)
    c = Parameter(index=(c, s), value= c̃_cs)
    a = Parameter(index=(c, k), value= a_ck)

    # 変数
    x = BinaryVariable(index=(c,s))

    # 目的関数
    problem += Sum((d[c]+α*c[c,s])*x[c,s] )

    # 制約条件
    problem += Sum(x[c,s]*a[c,k] ,(s,k)) >= 1
    problem += Sum(x[c,s] ,c) <= 1

    problem.solve
```

5．計算実験（1/3）

■ EnumとEnum+に対する列挙の計算時間，集合被覆モデルの計算時間，２つの合計時間を
を以下に示す

■ 計算環境

CPU: Intel Core i9-9900K CPU @ 3.60 GHz　　Memory: 32 GB

Solver: Numerical Optimizer V23（PySIMPLE 1.2.1）　Python: Python 3.9

合計工事難易度上限：9　重み係数α：1

工事数[件]	人数[人]	枠数[件]	最適解	Enum			Enum+		
				列挙	集合被覆	合計時間	列挙	集合被覆	合計時間
10	4	3	4,360	0.00	0.99	0.99	2.65	0.06	2.71
10	4	4	4,017	0.00	25.01	25.01	4.74	0.20	4.94
10	4	5	3,874	0.01	112.81	112.82	5.72	0.24	5.96
10	4	6	3,874	0.00	3.18	3.18	5.76	0.27	6.03
10	4	7	3,874	0.02	3.15	3.17	5.76	0.27	6.03
10	4	8	3,874	0.01	3.20	3.21	5.85	0.27	6.12
12	5	3	5,935	0.00	4.07	4.07	4.54	0.13	4.67
12	5	4	5,194	0.02	4.01	4.03	9.93	2.41	12.34
12	5	5	5,046	0.03	24.33	24.36	12.17	3.78	15.95
12	5	6	5,046	0.03	31.38	31.41	12.34	3.10	15.44
12	5	7	5,046	0.04	31.77	31.81	12.31	3.09	15.40
12	5	8	5,046	0.04	31.52	31.56	12.21	3.14	15.35
13	5	3	6,824	0.00	6.16	6.16	5.67	0.28	5.95
13	5	4	6,356	0.03	7.31	7.34	12.67	3.87	16.54
13	5	5	6,249	0.04	39.92	39.96	15.59	5.48	21.07
13	5	6	6,249	0.04	48.93	48.97	15.54	5.83	21.37
13	5	7	6,249	0.05	49.10	49.15	15.80	5.84	21.64
13	5	8	6,249	0.05	49.39	49.44	15.36	5.76	21.12
15	6	3	6,344	0.01	7.10	7.11	8.67	1.81	10.48
15	6	4	5,650	0.03	24.45	24.48	19.55	11.45	31.00
15	6	5	5,647	0.06	117.80	117.86	23.20	9.25	32.45
15	6	6	5,647	0.06	135.87	135.93	23.46	15.52	38.98
15	6	7	5,647	0.07	138.21	138.28	23.48	15.53	39.01
15	6	8	5,647	0.07	139.97	140.04	23.78	15.47	39.25

－: time out (36,000sec)

5．計算実験（2/3）

■ EnumとEnum+に対する列挙の計算時間，集合被覆モデルの計算時間，２つの合計時間を
を以下に示す

■ 計算環境

CPU: Intel Core i9-9900K CPU @ 3.60 GHz　　Memory: 32 GB

Solver: Numerical Optimizer V23（PySIMPLE 1.2.1）　Python: Python 3.9

合計工事難易度上限：9　重み係数α：1

工事数[件]	人数[人]	枠数[件]	最適解	Enum			Enum+		
				列挙	集合被覆	合計時間	列挙	集合被覆	合計時間
18	7	3	7,363	0.01	36.21	36.22	14.92	9.84	24.76
18	7	4	6,889	0.08	330.94	331.02	44.25	39.76	84.01
18	7	5	6,741	0.21	1,538.99	1,539.20	62.10	25.08	87.18
18	7	6	6,741	0.31	3,357.60	3,357.91	62.97	33.90	96.87
18	7	7	6,741	0.34	3,392.29	3,392.63	63.34	34.10	97.44
18	7	8	6,741	0.35	3,649.04	3,649.39	63.65	33.80	97.45
20	8	3	7,230	0.01	55.01	55.02	20.54	11.86	32.40
20	8	4	6,856	0.14	1,092.49	1,092.63	68.40	84.69	153.09
20	8	5	6,626	0.45	7,028.75	7,029.20	105.70	50.16	155.86
20	8	6	6,626	0.86	28,225.42	28,226.28	113.93	56.24	170.17
20	8	7	6,626	—	—	—	114.34	55.83	170.17
20	8	8	6,626	—	—	—	114.96	57.65	172.61
36	14	3	14,251	0.11	1,823.90	1,824.01	123.88	293.60	417.48
36	14	4	13,727	—	—	—	597.59	6,570.69	7,168.28
36	14	5	13,387	—	—	—	1,002.68	18,827.78	19,830.46
36	14	6	13,387	—	—	—	1,125.96	116.04	1,242.00
36	14	7	13,387	—	—	—	1,126.20	23,735.01	24,861.21
36	14	8	13,387	—	—	—	1,237.98	24,484.67	25,722.65
40	15	3	9,586	0.19	13,837.05	13,837.24	155.02	1,013.60	1,168.62
40	15	4	8,903	—	—	—	845.53	9,781.02	10,626.55
40	15	5	8,898	—	—	—	1,726.93	29,589.83	31,316.76
40	15	6	8,898	—	—	—	—	—	—
40	15	7	8,898	—	—	—	—	—	—
40	15	8	8,898	—	—	—	—	—	—

—: time out (36,000sec)

5．計算実験（3/3）

- 多くの問題例で厳密な最適解を得るまでの計算時間を大幅に削減した
- Enum+は列挙時に時間を要しているが，集合被覆モデルを解く時間は小さく，合計時間も小さくなった
- 問題例の規模や立会者の担当する工事件数が大きいほど， Enumより Enum+のほうが高速に厳密な最適解が得られた
- Enumでは厳密な最適解が得られない問題例に対して，Enum+では実用的な時間で厳密な最適解が得られた

6．まとめ

- 各立会者に割り当てる工事件数が大きい場合でも実用的な時間で良質な解を得られる手法を提案した
- 提案した手法により，実データをもとに生成した問題例に対し，実用的な時間で厳密な最適解が得られること計算実験により確認した
- 今後の課題は，近似解法による計算時間のさらなる短縮を検討することである

参考文献

[1] 国土交通省, インフラ老朽化対策　(平成27 年9 月11 日第2 回非社会保障ワーキング・グループ資料 1-3 国土交通省資料).
https://www5.cao.go.jp/keizaishimon/kaigi/special/reform/wg2/270911/agenda.html (Retrieved on October 13, 2019)
[2] 国土交通省, 浅層埋設にあたっての安全対策について(2015 年7 月31 日第5 回無電柱化低コスト手法技術検討委員会資料3 浅層埋設にあたっての安全対策について).
http://www.nilim.go.jp/lab/ucg/koho/k150731.html (Retrieved on October 13, 2019)
[3] NTT 東日本, 電話ケーブル切ったら大へん　(NTT 東日本東京事業部2018).
http://kirenkyo.gr.jp/sites/default/files/doc/NTT higashi2018.pdf (Retrieved on October 13, 2019)
[4] NTT 西日本, 管路・電柱等.
https://www.ntt-west.co.jp/open/99guidebook/pdf/2-6syo.pdf (Retrieved on October 13, 2019)
[5] 月刊ビジネスコミュニケーション, ワンストップサービスを提供するインフラネットのIT システム群.
https://www.bcm.co.jp/magazine/00-02/html/052.html (Retrieved on October 13, 2019)
[6] S. Umetani, M. Arakawa and M. Yagiura, Relaxation heuristics for the set multicover problem with generalized upper bound constraints, Computers & Operations Research, 93 (2018), 90–100.
[7] 高須賀将秀, 柳浦睦憲, 工事手配業務に対する数理最適化の活用と意思決定の支援, NTTデータ数理システム, 2020年春・学生研究奨励賞.
https://www.msi.co.jp/nuopt/download/stuAward/module/No2_muc20_NUOPT.pdf (Retrieved on November 27, 2021)
[8] 高須賀将秀, 柳浦睦憲, 工事手配業務に対する数理最適化の活用と意思決定の支援, 情報処理学会論文誌数理モデル化と応用, 14 (2021), 112–120.
[9] Dantzig, G., D. Fulkerson, S. Johnson, Solution of a large scale traveling salesman problem, Operations Research, 2 (1954), 393–410.